Conditional Generation of
Temporally-ordered Event Sequences

APPROVED BY

SUPERVISING COMMITTEE:

Greg Durrett, Supervisor

Katrin Erk

Conditional Generation of Temporally-ordered Event Sequences

by

Shih-Ting Lin

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Computer Science

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2020

Acknowledgments

I wish to acknowledge everyone who has played a role in my academic accomplishments. First of all, my parents, who supported me for love and understanding. Secondly, my supervisor and committee members, each of whom has provided valuable advice and guidance throughout the research process. Thank you all for the tremendous support.

Conditional Generation of Temporally-ordered Event Sequences

Shih-Ting Lin, MSCompSci
The University of Texas at Austin, 2020

Supervisor: Greg Durrett

Narrative schema knowledge is useful for a wide range of tasks related to events. In this work, we tackle the problem of making inferences from *partial*, *unordered* sequences of events, which arise from having incomplete knowledge about some underlying scenario. We tackle the problems of temporal event ordering and event infilling, predicting new events which fit into a sequence of existing ones according to temporal and coherence criteria. We unify these problems in a single model, a BART-based conditional generation model learned as a denoising autoencoder. This model operates over sequences of events represented as predicate-argument structures. At training, we take event sequences, shuffle them, delete events, and then attempt to recover the original events to encourage our model to capture more general temporal event knowledge. Our evaluation demonstrates that our BART-based models significantly outperform both a BERT-based pairwise model and a BERT-based pointer network on temporal event ordering. A human evaluation also shows

that our models are able to generate events that fit more temporally into the input events compared to GPT-2 models.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Learning of narrative schemas is now a well-studied problem in NLP [27, 3, 35, 36]. However, how a schema should be represented (discretely or with distributed representations?) and exactly what inferences we want to make with these is not always clear. Downstream applications can give us some clues. Specifically, understanding typical temporal orders of event sequences is an important component in certain question answering settings [60, 32, 61]. In text generation, story completion relies on understanding what makes narratives plausible and what events might be likely to happen before, after, and between other events [15, 57]. These applications indicate the need to build a temporal-aware model which supports two tasks: temporal event ordering and event infilling, or inferring unseen or unmentioned events occurring as part of a larger scenario.

To obtain a model that can capture temporal knowledge to solve both event ordering and infilling, there are two main research questions we need to answer. First, we need a suitable training scheme that can provide supervision signal to encourage our model to *truly* capture strong temporal knowledge about events. This kind of knowledge is required for schema learning, but

may not necessarily be learned by traditional in-context temporal ordering models which rely on discourse information [6, 31]. Second, we ideally want to be able to address multiple temporal ordering tasks with a single model architecture, rather than having to annotate data and build ad hoc models for tasks like temporal ordering and event infilling.

In this work, we propose a conditional generation model with a denoising training scheme to tackle these two problems. Specifically, we formulate this problem as a denoising autoencoder over event sequences that we will tackle with a pre-trained sequence-to-sequence model. The encoder of our autoencoder reads a partial, potentially temporally scrambled sequence of input events. The decoder of our autoencoder reconstructs the original event sequence in temporal order; this can be viewed as a conditional event language model [18, 1, 24]. This denoising task can help models to learn temporal event knowledge that we can then apply in various ways to solve downstream tasks. Such denoising training has been successful exploited in many applications [49, 22, 19], but to our knowledge we are the first to apply it in this temporal modeling setting.

The conditional generation architecture of our model also makes our framework flexible enough to solve the two targeted event ordering and event infilling tasks. We can frame both tasks as conditional generation and either sample from our model or score different sequences using it. Given the recent success of pretrained of encoder-decoder transformers [21, 43], in this paper we proposed two variants of a novel BART-based conditional generation model.

These two models are designed to use different input and output formats, but both allow us to adopt the text-to-text transformer to event-based generation tasks.

Gathering large-scale high-quality labeled data with temporal annotations is often expensive and requires specially designed annotation schemes [39, 2, 31]. Here, we instead turn to a narrative documents corpus, EventsNarratives [58] and design an automatic method to extract the training data we need. In these documents, discourse order can be assumed to reflect temporal order, so they are an ideal source of training data for our models.

We evaluate the capability of our proposed models on solving the two targeted tasks by applying them on out-of domain test sets in a zero-shot manner. Specifically, for event ordering, we first extract test temporal event sequences from CaTeRS and MCTaco dataset, which include the annotations on temporal relations between events. We then compare the performance of our models with two baselines: a BERT-based pairwise model and a BERT-based pointer network. For event infilling, we reuse the test event sequences from CaTeRS, and examine the ability of our models on tackling unseen events, which is useful for event infilling, and actually generating infilled events in comparison with GPT-2 baselines.

Automatic evaluation results shows that our BART-based models significantly outperform the baselines models on both temporal event ordering and the ability to tackle unseen events. Human evaluation also verifies that our models can generate infilled events that are more temporally ordered with

respect to the input events.

Chapter 2

Related Works

2.1 Temporal Event Ordering

Closely related to our ultimate goal of schema learning and temporal modeling is the task of temporal relation extraction, which focuses on ordering pairs of events in text [40, 2, 31]. The early works in this area tackle this task as pairwise classification problem [25, 48, 5, 47, 7, 45]. However, this pairwise formulation could result in conflicting temporal predictions. One line of work addresses this issue by exploiting ILP and structure learning to enforce the structure constraints on the outputs [10, 30, 20, 13, 14], but the underlying models are still designed for leaning pairwise relations, which could hinder the model from learning to exploit non-local information. In contrast, in this work we view temporal event ordering as a sequence generation problem, which provides our models a stronger inductive bias to capture global temporal relations between events. One recent work [23] treats this task as a graph generation problem, and so is able to predict more complex structures, but it focuses on ordering and is not suitable for our ultimate goals.

2.2 Schema Induction

To achieve schema induction, a standard approach is learning schema-like event knowledge and use it to perform events prediction. To do so, a line of previous work attempts to use statistical methods to acquire useful event-related information [4, 16, 27]. Another line of work exploits event language modeling to learn the distributions over events [38, 35, 55], or focuses on learning event representations [26, 53] rather than writing down discrete schemas. However, most of these works only model the co-occurrence between events instead of directly considering temporal information, and only represent events in S-V-O format, which doesn't include rich enough information for the model to well capture the temporal event knowledge.

Another line of work instead directly focuses on narrative generation, which is one application of schema induction. Wang et al. [50] particularly aims at extracting coherent narratives from story salads. Other works further attempt to address the task of generating narratives given predefined scenarios [51, 41]. However, these do not address the task from a temporal ordering standpoint. Without considering the temporality, these works are more prone to learn discourse structures instead of the temporal knowledge that can be applied to schema learning.

Chapter 3

Method

In this section, we describe our proposed conditional generation model along with the event-based denoising autoencoder training scheme. This framework allows us to effectively capture temporal event knowledge and use it to tackle both event ordering and event infilling with a single model.

3.1 Denoising Autoencoder Framework

We define our task as a sequence-to-sequence problem. Our model takes as input a sequence of events $\mathbf{x} = \{e_1, \cdots, e_m\}$; these should be part of the same scenario and involve shared actors, but are not necessarily in temporal order. For the definition of events, here we follow Chambers and Jurafsky [4] in general and consider an event e to be a predicate v_e along with its arguments from the sentence $\mathbf{s}_e$ which v_e lies in. We then place a distribution over related sequences of events $\mathbf{y} = \{e_1, \cdots, e_l\}$ according to a distribution $P(\mathbf{y} \mid \mathbf{x})$. The output sequence of events should be related to the input events, coherent, and most importantly **temporally ordered**: if the input is a partially-ordered, messy collection of information, the output should represent distributions over a true underlying order of events, without obvious gaps in the event sequence.

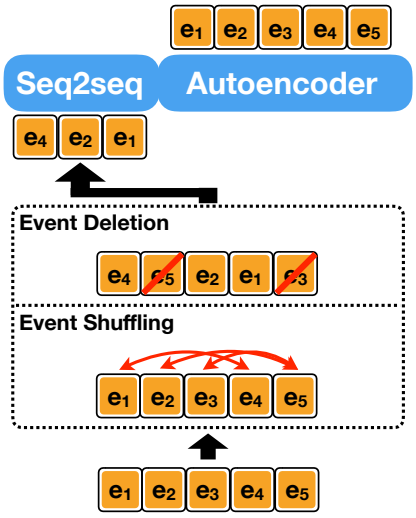


Figure 3.1: An overview of the event-based denoising autoencoding training scheme used to encourage our model to learn temporal event knowledge. Two transformation functions are used to corrupt the input event sequences: event deletion and shuffling.

Note that compared to past work tackling similar scenarios like that on story salads [50, 51], we deal with incomplete events and focus on temporal order rather than discourse order. Finally, note that it is not a strict requirement that all input events be contained in the output sequence, though that is the case for the training data we construct in this work.

We train this model as an event-based denoising autoencoder. The training scheme of a denoising autoencoder framework is composed of two steps. Specifically, given a sequence of temporally ordered events $\mathbf{x} = \{e_1, \cdots, e_m\}$, we first corrupt it by a predefined transformation function into $\mathbf{x}' = \{e'_1, \cdots, e'_n\}$. We then learn our model to maximize $P(\mathbf{x} \mid \mathbf{x}')$. To guide the model to capture temporal information targeted towards downstream event ordering and

event infilling tasks, for corrupting the input, we exploit two transformation functions, as shown in Figure 3.1:

Event Shuffling This function performs a random shuffling over the events in $\mathbf{x}$ to produce $\mathbf{x}'$. To perfectly reconstruct the original event sequence $\mathbf{x}$, the model is required to learn the temporal relations between events. Additionally, with the random shuffling, we can treat this denosing task as predicting the temporally ordered event sequence given a unordered event set, which is similar to the goal of event ordering. This suggests that the shuffling denoising scheme can help our model to learn temporal event ordering.

Event Deletion This function deletes each event in $\mathbf{x}$ randomly with a probability p to produce $\mathbf{x}'$. This denoising scheme is similar to token deletion transformation introduced in Lewis et al. [21]. To perfectly reconstruct the original event sequence, the model need to encode the schema-like event knowledge so that it can generate events not included in the input $\mathbf{x}'$ and insert them at correct positions. As a result, this denoising task can help the model to learn event infilling, which has a similar goal.

The final denoising transformations we use to construct training data for our model is the combination of the above two functions. Concretely, given an event sequence $\mathbf{x}$, we first perform event shuffling, then randomly delete events from it to produce $\mathbf{x}'$.

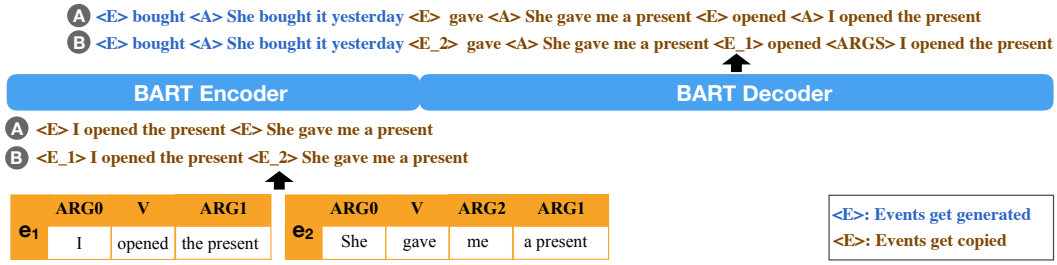


Figure 3.2: The overall model architecture of the proposed BART-based conditional generation models, which takes events (predicates for this work) with their arguments as input. Two input and output formats are designed to solve the task of event-based denoising autoencoding. A: The EventBART model which prepends the same special event tokens for all of the events. B: The EventBART-indexed model which instead uses indexed special event tokens. In this figure, we use `<E>` to represent `<EVENT>` and `<A>` for `<ARGS>` for convenience.

3.2 Model Architecture

We build our autoencoder with a sequence-to-sequence model. Specifically, to leverage the power of pretrained transformers, we adopt BART [21] as the underlying architecture, and initialized our model with its pretrained weights.

The overall model is shown in Figure 3.2. To fit the event-based inputs and outputs to BART, we need to represent each event e with a textual format Repr(e). To achieve this, we represent an event e with the concatenation of its predicate and arguments [33] extracted by applying a SRL model on $\mathbf{s}_e$. Some previous works instead extract the syntactic heads of arguments of v_e and represent e as an n-tuple of the head tokens, where n is normally 3 to 5 [37, 53, 54]. However, this design limits the information we can pass to the

model to learn the temporal knowledge. Another line of works feed the entire sentence $\mathbf{s}_e$ into the model [13, 14]. This approach, however, can result in a model that learns to rely on discourse clues or other biases in the dataset, especially when solving the event ordering task. Our representation strikes a balance between the oversimplified tuple-based representation and the entire sentence which tends to include extra information.

With this SRL-based event representation, we consider two variants for the input and output format:

EventBART As mention in and §3.1, our BART-based autoencoder model is required to take a corrupted event sequence $\mathbf{x}' = e'_i$ as input, and output another event sequence $\mathbf{y} = e_j$. To feed $\mathbf{x}'$ into the model, we first encode each event e'_i as $\text{Repr}(e'_i)$, and concatenate them with a special event token `<EVENT>` prepended in front of each event. This special token can help the model to identify the boundary between the input events. As for the output, we instead prepend `<EVENT>` v_{e_j} `<ARGS>` in front of each $\text{Repr}(e_j)$. This setup not only provides an extra supervision signal that encourages the model to perform predictions on the basis of *events*, but also allows us to match the events more easily by checking the predicates part.

EventBART-indexed In this model, the input and output format is the same as that in the EventBART, except the prepended special tokens `<EVENT>`. Here, as shown in Figure 3.2, for the input we instead prepend `<EVENT_i>` be-

fore each event e'_i. For the output, if e_j is one of the input events and $e_j = e'_i$, then we also changed the prepending tokens e_j to <EVENT_i> v_{e_j} <ARGS>. Otherwise, we still use <EVENT> as the special event token. Note that the model is *not* able to "cheat" using the <EVENT_i> tokens to do the prediction since the input events are scrambled by the shuffling denoising training scheme described in §3.1. This setting has two advantages comparing to the EventBART for the event ordering task. First, the use of <EVENT_i> provides a clue for the model to associate input events to output events, which can benefit the event ordering. Second, in event ordering, what we want ideally is to produce event sequences by only judging whether they are temporally ordered. However, in EventBART, we are forced to include the correlation of textual arguments between events, or even within events in the ordering score. The use of <EVENT_i> provides us a work-around. Specifically, when scoring whether a output sequence $\mathbf{y}$ is temporally ordered, instead of using the generation probability on the entire text representation of $\mathbf{y}$, $\{w_t^{\mathbf{y}}\}$:

$$P^{\text{gen}}(\mathbf{y}|\mathbf{x}') = \prod_t \text{BART}(w_t^{\mathbf{y}}|w_1^{\mathbf{y}}, \cdots, \mathbf{x}') \tag{3.1}$$

we can gathered the generation scores on the special tokens <EVENT_i> only as its final ordering score:

$$P^{\text{tag}}(\mathbf{y}|\mathbf{x}') = \prod_{t \in \mathbf{I}} \text{BART}(w_t^{\mathbf{y}}|w_1^{\mathbf{y}}, \cdots, \mathbf{x}') \tag{3.2}$$

where $\mathbf{I}$ is the set of the positions of the special tokens <EVENT_i> in $\{w_t^{\mathbf{y}}\}$. This allows us to make a judgment only depend on the predicted temporal ordering

of events. In our evaluation on event ordering task, we also experiment with this method, which is denoted as "EventBART-indexed (tags only)".

3.3 Training Data Collection

For our framework, the training data we need is event sequences *in temporal order*. Note that most text data occurs in *discourse order*, which is not the same thing: frequently an event mentioned before another one will occur first, but not always, as we can see from the human annotations in temporal relation datasets such as TimeBank [40].

Existing datasets [2, 46] are small-scale, and annotating more data is expensive and prone to low agreement [31]. To combat this issue, we instead try to automatically gather the training data we need.

Corpus We focus on the EventsNarratives corpus [58], and try to automatically collect temporal event sequences from each document within. EventsNarratives is a corpus consists of more than 200,000 *narrative-structured* documents, which are identified from three different source domains including news articles, novel books and blogs using a weakly supervised approach. Yao and Huang [58] use a method to identify narrative texts, describing a sequence of events in such a way that the discourse order is very likely to reflect the temporal order. This gives us an entry point to collect temporal event sequences automatically. Here we focus on documents in the novel domain as our source for temporal event sequences.

Extracting Temporal Event Sequences To obtain the training event sequences, we first use an SRL model from AllenNLP [11] to extract verbs and their arguments, and view those verbs as events. Then, temporal event sequences are constructed by connecting only the events in different sentences, since the relations between events within the same sentence are unclear even in narrative documents. Here, to ensure all the events in a sequence have a strong relation with each other, we only include chains of events **that are associated with a common entity**, as determined by checking whether the arguments of two event have some shared non-stopped tokens. With this procedure, we are able to collect nearly 2 million temporal event sequences as our training set, with nearly 70% of the sequences consisting of three or more events.

3.4 Training Details

We train our BART-based conditional generation models using negative log likelihood as the reconstruction loss. We set the learning rate to 1e-5, and use a polynomial decay scheduling with 500 steps of warm-up. All of the models are trained for 10 epochs, with each epoch being 2000 updates and the batch size being 64. For the deletion training scheme, we set the event deletion probability p to 0.15. The framework is implemented with PyTorch [34] and AllenNLP [11], and we use the BART-large pretrained model from HuggingFace's Transformers library [56].

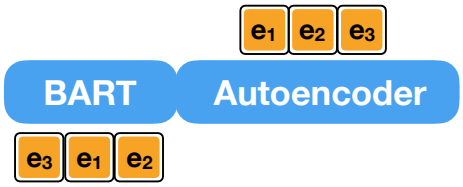
Chapter 4

Targeted Tasks Formulation

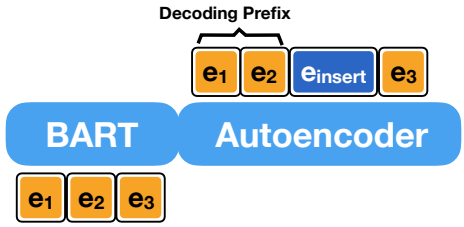
Here we elaborate the formulation of the two targeted tasks of our model: temporal event ordering and event infilling. Since in general, these two tasks can be seen as event-based conditional generation problems, they are both solvable in an end-to-end manner by our BART-based autoencoder model. An overview of the task formulations are shown in Figure 4.1.

4.1 Temporal Event Ordering

Given an unordered set of events $\{e'_i\}$, the goal of this task is to produce the temporal ordering of the input events. Unlike one line of works in this area which address this task by pairwise classification, we instead ask the model to generate an ordered sequence of events $\{e_{f(i)}\}$ given the set $\{e_i\}$, where $f(\cdot)$ is a mapping function to determine the event to put at the ith position. This is a conditional generation problem that can be directly solved by our proposed models.



(a) Temporal Event Ordering: given a temporally scrambled event sequence $\{e_3, e_1, e_2\}$, we use our BART model to generate its temporally ordered permutation



(b) Event Infilling: given a seed event sequence $\{e_1, e_2, e_3\}$ and a pre-selected insertion position (i* = 3 here), we use our BART model to generate an inserted event (highlighted in blue) with $\{e_1, e_2\}$ as the decoding prefix

Figure 4.1: An overview of the formulation of the two targeted tasks in this work. (a) illustrates the temporal event ordering task, and (b) illustrates the event infilling task.

4.2 Event Infilling

The goal of event infilling is to generate inserted events at some pre-selected insertion positions in a seed event sequence [52]. To simplify the evaluation, here we assume that given an event sequence $\mathbf{x} = \{e_i\}$, models will only be required to generate *one* inserted event at *one* insertion position i^*. This task can also be tackled by our models. Specifically, we first feed $\{e_i\}$ as the input to the model. During decoding, we then ask the model to generate one event e_{insert} using $\mathbf{x}_{\text{prefix}} = \{e_i | i < i^*\}$ as the decoding prefix.

The rest of the events in $\mathbf{x}$ are directly after the inserted event if we need the complete resulting event sequence. To force our models to produce $e_{\text{insert}} \notin \mathbf{x}$, we directly block models to generate $\{v_{e_i}\}$ during the decoding process.

Chapter 5

Baselines

In this section, we introduce the baselines we use to evaluate the performance of the proposed BART-based models on our two targeted tasks .

5.1 Temporal Event Ordering

This task is concerned with taking a set of events $\mathbf{x}' = \{e'_1, \cdots, e'_n\}$ (possibly in random order) and putting them in the correct temporal order. The output $\mathbf{y}$ is the specific permutation of the input events that makes it temporally ordered. Note that unlike past systems trained on datasets like Timebank, we are not doing *in-context temporal ordering* where we see the events in the context of the original discourse. Instead, we consider a more flexible version of this task that could potentially be applied to events drawn from different texts and settings where temporal ordering is not a strong cue.

BERT-based Pairwise Model + SSVM We follow the architecture of the Deep SSVM model used in Han et al. [13] as our first baseline, which tackles event ordering as a pairwise classification problem. This network first exploits a BERT-based model [9] to obtain the vectorized representation for each input

event e'_i in $\mathbf{x}'$. Formally, same as the BART-based models, the input to the BERT model is the concatenation of Repr(e'_i) with `<EVENT>` being prepended in front of each event. The vectorized representation for e'_i is then extracted by $\mathbf{U}_{p_i}$, where $\mathbf{U}$ is the final BERT encoded matrix, and p_i is the position of the first token of e'_i in the input sequence. Each pair of event representations, $\mathbf{U}_{p_i}$ and $\mathbf{U}_{p_j}$ are then fed into a feed-forward function g to compute a score B for e'_i preceding e'_j in the output $\mathbf{y}$:

$$B(e'_i, e'_j) = g([\mathbf{U}_{p_i}; \mathbf{U}_{p_j}; \mathbf{U}_{p_i} \odot \mathbf{U}_{p_j}]) \tag{5.1}$$

Finally, the final output $\mathbf{y}$ is computed by performing an ILP over all of the pairwise scores. The overall network is trained with the structure SVM loss so that it can learn to make joint predictions with transitivity constraint. We denote this baseline as "Pairwise+SSVM" in the evaluation section.

BERT-based Pointer Network This network first follows BERT-based Pairwise Model + SSVM to extract the the vectorized representation $\mathbf{U}_{p_i}$ for each e'_i. These event representations are then instead fed into a LSTM-based pointer network to model the ordering probability by decomposing it in a sequential decoding manner:

$$P^{\text{seq}}(\mathbf{y}|\mathbf{x}') = \prod_j P(j|\mathbf{h}_1, \cdots, \mathbf{U}_{p_1}, \cdots) \tag{5.2}$$

$\mathbf{h}_t$ is the decoder hidden states in the pointer network. Comparing to the above pairwise baseline, this pointer network model has a stronger inductive bias for exploiting global event relations. We train the sequential model with

teacher-forcing to maximize the probability of the gold ordering. We denote this baseline as "BERT-based PN" in the evaluation section.

5.2 Event Infilling

GPT-2 GPT-2 [42] is a transformer-based pretrained language model with unidirectional decoding architecture. This model is effective at various generation tasks such as story generation [8, 44]. However, one issue with the GPT-2 model is that it can only perform uni-directional generation. To make it work on event infilling, we use the GPT-2 model to generate the infilling event e_{insert} conditioned on $\mathbf{x}_{\text{prefix}}$ only. Specifically, similar to BART-based models, we first transform each event e_i to $\text{Repr}(e_i)$ appended with a period, then concatenate them together as the decoding prefix for our GPT-2 baseline. After the decoding process, a predicted text segment will be generated, which should be a plausible narrative following the prefix events. We then split the text segment by periods, and use the first split as the text representation of e_{insert}, which is used to judge the infilling quality. We use the GPT2-medium pretrained model from HuggingFace's Transformer [56], whose model size is comparable to BART-large, if not specified otherwise.

Infilling GPT-2 One clear downside of the GPT-2 baseline is that it can only condition on limited input events $\mathbf{x}_{\text{prefix}}$ to generate the infilling events. To build a stronger GPT-2 baseline, we follow the baselines from Qin et al. [41] to exploit GPT-2 on infilling tasks. Specifically, infilling GPT-2 generates the

infilling events conditioned on the full input events by "wrapping" the events after the insertion position to the front. That is: the decoding prefix fed to the infilling GPT-2 becomes the concatenation of $\{\text{Repr}(e_i)|i >= i^*\}$, `<SEP>` and $\{\text{Repr}(e_i)|i < i^*\}$, again with a period appended after each event. The special token `<SEP>` is used to help the model to differentiate the events before and after the insertion position.

Chapter 6

Evaluation

We now evaluate the capability of our proposed model in capturing required temporal event knowledge to tackle the two targeted tasks: temporal event ordering and event infilling.

6.1 Experiment Setup

All the models used in the evaluation are trained with the temporal event sequences automatically collected on EventsNarratives except GPT-2, since we want to compare the learned knowledge in GPT-2 with our proposed models. Although we are able to gather millions of sequences, for efficiency, we train on 100,000 sequences unless specified otherwise. For each sequence, we extract 2 distinct permutations from the corruption process. This results in 200,000 training examples in total.

During evaluation, all the models are evaluated on out-of-domain datasets in a zero-shot way, i.e., no fine-tuning is performed on the evaluation sets. These downstream tasks will then directly evaluate the degree of temporal event knowledge present in our model and whether it can be adapted to solve the targeted tasks, which is the main goal of this work.

6.2 Temporal Event Ordering

We first perform evaluations on the temporal event ordering task, whose goal is to produce the temporal order of a set of input events. For the BERT-based pointer network and both of the BART-based models, we directly generate the ordered event sequences with beam search with beam size 4.

6.2.1 Datasets

There are two out-of-domain datasets being used here for extracting the test temporal event sequences: CaTeRS and MCTaco. Same as during training, two different permutations are produced from each extracted sequences.

CaTeRS [29] CaTeRS is a human-labeled dataset developed for script learning. It includes the annotations of events and their casual and temporal relations on 320 five-sentence short stories sampled from ROCStories corpus [28]. To extract the temporal event sequences from CaTeRS, for each storty, we first apply the same SRL model used in §3.3. Then, an acyclic directed graph is constructed with a node being an event e whose predicate v_e can be captured by the SRL model, and an edge (e_i, e_j) indicating e_i happens temporally before e_j. Note that here we treat all types of annotated relations except "IDENTITY", "DURING" and "CAUSE_TO_END" as "BEFORE", as suggested in Mostafazadeh et al. [29]. Test temporal event sequences are then extracted by retrieving all the path from the source nodes to sink nodes in the graph. With this procedure, we are able to gathered 842 event sequences, 60% of which

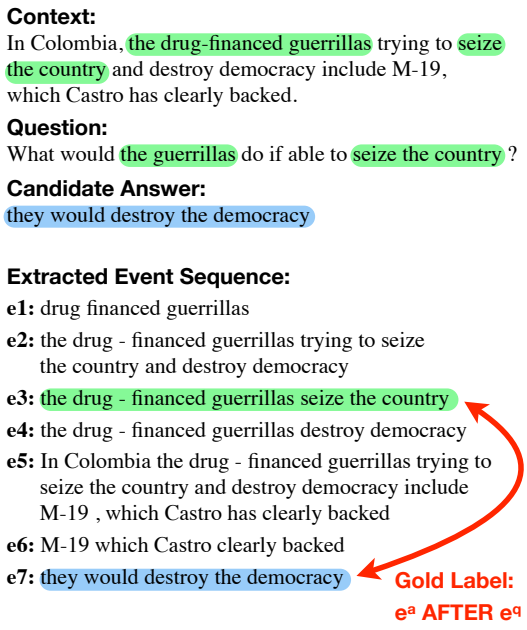


Figure 6.1: An example of the event sequence extracted from a context-question-answer tuple in MCTaco. e^q and e^a are highlighted with the color green and blue respectively.

contain 3 or more events. With permutations, the final CaTeRS evaluation set has 1684 examples.

MCTaco [60] MCTaco is a multiple-choice question answering dataset designed to evaluate models's understanding on 5 different types of temporal commonsense. To extract suitable test data, we resort to questions with the reasoning type of "event ordering" and their positive candidates. Each data point here is a (sentence, question, candidate answer) tuple, with the sentence describing multiple events and the question asking about whether the candidate answer, which is also an event, happens temporally before/after a

Architecture	All Pairwise Acc.	Length >= 3 Pairwise Acc.
Pairwise+SSVM	65.7	62.3
BERT-based PN	54.1	52.3
EventBART	77.1	74.7
EventBART-indexed	**79.7**	**78.0**

Table 6.1: The averaged pairwise accuracy between the gold ordering and predicted ordering generated by each model on temporal event sequences extracted from CaTeRS dataset. The numbers on the right super column are computed only with the test sequences with 3 or more events. Both of our BART-based models significantly outperform the two baselines.

particular event in the sentence. Our steps for extracting test data from a MCTaco tuple is as follows: First, we apply a SRL model on the sentence to extract a set of events $\{e_i^c\}$. The event in the answer e^a is also extracted by the SRL model. For the question, we first use a set of self-defined regex template to extract an event e^q and a temporal relation ("before" / "after'). We then match e^q to one of e_i^c by ROUGE-L scores. Critically, *the question itself* tells us whether e^a should be before/after e^q in the temporal event sequence formed by $\{e_i^c\} \cup \{e^a\}$. With this annotation, the way we evaluate our models here is: we first feed the randomly shuffled $\{e_i^c\} \cup \{e^a\}$ into a model, which will then produced an ordering. We then check whether e^a is indeed before/after e^q in the predicted ordering. The number of test sequences we are able to extract from MCTaco is 585 in total. Comparing to CaTeRS, since the sentences here are from 9 different domains in MultiRC [17], the type of events are more diverse. The arguments of events are also more complex. An example of the extracted data is shown in Figure 6.1.

6.2.2 Results on CaTeRS

We first examine the evaluation results on CaTeRS, which are shown in Table 6.1. To quantitatively judge the performance of each model, here we compute the pairwise accuracy, which represents how many pairs of events within the input are ordered correctly by a model. For the BART-based models, since there is no guarantee to generate permutations of the input event, we additionally match the events between the output and input using the predicates and the special event tokens, which exist in both the input and output event format, before computing the pairwise accuracy.[1]

Our BART-based models outperform the BERT-based pointer network by more than 20 points, a huge margin. One possible reason is that the decoder of BART can condition on the token-leveled embeddings of the events when generating the output events, whereas in the pointer network, the decoder is only aware of the condensed event embeddings $\mathbf{U}_{p_i}$.

Our two BART-based models also outperform the BERT-based Pairwise model, which is an architecture used in Han et al. [13]. This indicates our seq2seq model can better learn and exploit temporal knowledge to solve the event ordering. If we compare models' performance on the long sequences, we can see that our models suffers a lower drop compared to the baselines: while PAirwise+SSVM gets a 3.4-point drop, the decreases in our models are within

[1]The averaged matching F1 score of the output events generated by our models with respect to the input events is over 98, which indicates that our models are actually good at generating permutations of the input sequence.

Architecture	Acc.	Macro F1
Majority	90.6	47.5
Pairwise+SSVM	67.2	47.0
BERT-based PN	54.7	42.7
EventBART	63.9	50.1
EventBART-indexed	**74.9**	**55.1**
EventBART-indexed (tags only)	**76.6**	**56.4**

Table 6.2: The result of temporal event ordering on the event sequences extract from the MCTaco dataset. Metrics are computed on the temporal ordering between the answer event and sentence event. Note that the test set here is largely imbalanced, so we also include macro F1 in our metrics. Our BART-based models outperform the neural baselines as well as the majority baseline in terms of the macro F1, with using generation scores only on the special tokens further boosting the performance.

2.5 points. This result suggests that seq2seq modeling could benefit event ordering by encouraging the model to leverage non-local temporal relation between events, while in pairwise models ordering predictions are constructed by pairwise decisions with ILP to enforce structure constraints post-hoc.

6.2.3 Results on MCTaco

Next, we examine the evaluation results on MCTaco, which are shown in Table 6.2. Here since we only know the gold temporal relation of one pair of events in the input, i.e e^q and e^a, the averaged accuracy on predicting the order of e^q and e^a is computed. In addition, since the ratio of before/after questions is significantly unbalanced in MCTaco, with 90% asking about the "after" relationship, we also compute the macro F1 score as our metric (averaging F1 across these two classes).

From the result, we can see that the two baselines only gets a F1 score similar to or lower than just picking the majority label. This is possibly due to the high diversity of events in MCTaco, which makes it much harder to apply model here in a zero-shot way. In contrast, the EventBART model achieves an F1 score about 3 points higher than the Pairwise+SSVM baseline. The EventBART-indexed performs even better in terms of both the accuracy and the macro F1, which shows that using the indexed special event tokens can help the BART model to better solve the event ordering task. Finally, here we also experiment with the EventBART-indexed (tags only) model mentioned in §3.2. This variant further boosts the performance of the EventBART-indexed by 1.3 point on the macro F1. This result verifies that this setting could help the model to prevent the ordering scores from getting affected by the text correlation, which is a serious issue in MCTaco since the arguments of events here are more complex.

6.3 Ordering Unseen Events

We evaluate our BART-based models on an additional variant of this ordering problem that better tests their capability as *generative* models. Recall that in our setting previously, BART conditions on the complete (but possibly scrambled) sequence of events.

We now consider ordering an event in the decoder that the model *does not* condition on in the encoder. Concretely, for each temporal event sequence in CaTeRS, we randomly select one event e_{insert} to ask the model to insert, and

Architecture	All		Length >= 3	
	EM	Top2 EM	EM	Top2 EM
Random	34.1	69.5	23.7	48.7
GPT-2	35.2	68.4	22.6	48.2
Infilling GPT-2	38.8	73.5	26.3	55.4
EventBART	57.7	83.3	48.2	70.6
EventBART-indexed	**58.4**	**87.4**	**50.9**	**77.4**
EventBART-indexed ($p = 0$)	42.4	73.0	29.8	53.8

Table 6.3: Comparison of the ability to tackle unseen events between our BART-based models and various baselines on the CaTeRS dataset. The numbers on the right super column are computed only with the test sequences with 3 or more events. Our BART-based models are better at modeling temporal relationships between seen events and new unseen events.

treat the rest of the sequence as the seed input event sequence $\{e_1, \cdots, e_N\}$. Then we check whether a model can correctly determine where to insert e_{insert} into the seed input sequence. Specifically, for both the BART-based models and the GPT-2 baseline, we use the generation probability to rank event sequences $\{e_1, \cdots, e_{i^*-1}, e_{\text{insert}}, e_{i^*}, \cdots, e_N\}$ for i^* between 1 and $N+1$ (all possible insertion locations). For infilling GPT-2, we instead rank all the possible candidates by the generation probability of e_{insert} conditioned on $\{e_i | i >= i^*\}$ `<SEP>` $\{e_i | i < i^*\}$.

If a model correctly ranks the gold candidate higher, it indicates that it can model temporal relationships between seen events and new *unseen events* it may generate.

The results are shown in Table 6.3, where we compute the top1 and top2 exact match (EM) between the gold event sequence and the candidate sequence that gets the highest ranking score. GPT-2 performs only as good

as the random baseline; infilling GPT-2, which is designed to solve the event infilling task and trained on narrative data, achieves slightly better results but is still barely better than random. Our BART-based models are significantly better. The result that EventBART-indexed achieves the best result here again shows the benefit of using indexed special event tokens on helping the model to capture temporal relation of events. Finally we also performs an ablation study on the deletion scheme during training (Figure 3.1), which is shown on the last row in the table. Not using deletions during training (setting p to 0) causes a major drop in performance, by over 15 points of EM. Deletion denoising is therefore critical for the model to consider and model new events.

6.4 Event Infilling

6.4.1 Evaluation Setup

Given a temporal event sequence from CaTeRS, we first randomly remove one event at index i^* from it, and denote the resulting event sequence as $\{e_1, \cdots, e_N\}$. We then ask our models to generate one extra event e_{insert} at the position i^* so that $\{e_1, \cdots, e_{i^*-1}, e_{\text{insert}}, e_{i^*}, \cdots, e_N\}$ are still temporally ordered. The step of removing one event ensures that it is possible to at least infill an extra event into the input event sequence.

We examine the quality of the generated inserted events by performing human evaluation on Amazon Mechanical Turk (MTurk). Specifically, we randomly sample 30 examples from CaTeRS and have 5 raters to judge the coherence and temporality of the inserted event generated from each model

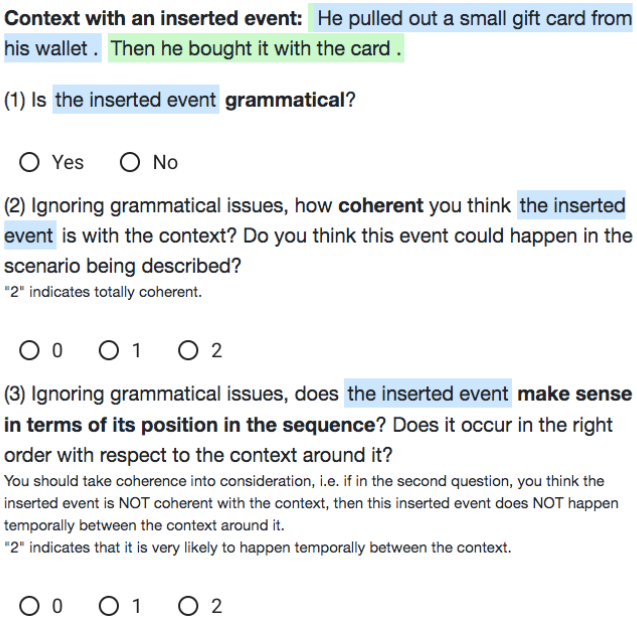


Figure 6.2: A screenshot of prompt for the human evaluation. The input events are highlighted with the color green, and blue for the inserted events. To help the raters to ignore grammatical issues when making decisions, we first ask them to check the grammaticality, then judge the coherence and the temporality.

on each example by asking the following two questions: (1) How coherent you think the inserted event is with the context in a scale from 0 to 2, if ignoring the grammatical issues? (2) Does the inserted event make sense in terms of its position in the sequence (in a scale from 0 to 2), if ignoring the grammatical issues? Our exact prompt can be seen in Figure 6.2. The final scores for each model on coherence and temporality are computed respectively by taking the average of the majority rating on each prediction.

Architecture	Coherence	Temporality
GPT-2	1.37	0.57
Infilling GPT-2	**1.50**	0.87
EventBART	1.43	**1.10**
EventBART-indexed	**1.50**	1.03

Table 6.4: The human evaluation on different sets of inserted event predictions for event infilling. The test data used here are the event sequences extracted from the CaTeRS dataset. Although all the models are able to generate coherent events, those produced by our BART-based models are more temporally ordered with respect to the input events.

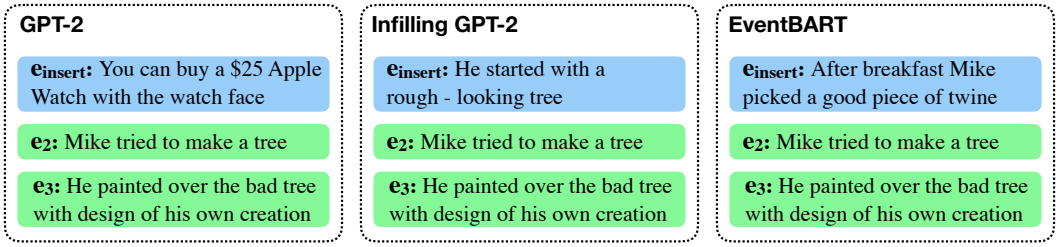


Figure 6.3: Examples of the infilled events generated by GPT-2, infilling GPT-2 and EventBART respectively. Note that these are real outputs from the three models. The events highlighted with the color green are the input events, and those highlighted with the color blue are the infilled events generated by the models.

6.4.2 Results

The results of human evaluation are shown in Table 6.4. For the quality of inserted events in terms of coherence, all the models get higher than 1 point on average, which means they all are effective at generating events that fit in the scenario described by the input events. In terms of temporality, GPT-2 performs worst among all models, as expected, since it can only conditioned on partial input event sequence while the other three are able to take the whole

event sequence as input. Furthermore, both of the BART-based models achieve a better performance than infilling GPT-2. The improvements on the temporal score are significant with $p < 0.05$ according to a bootstrap resampling test for both EventBART (0.034) and EventBART-indexed (0.041) with respect to infilling GPT-2.

In Figure 6.3, we also demonstrate examples of the infilled events generated by GPT-2, infilling GPT-2 and EventBART respectively. Given this specific test example, GPT-2 generates an event generally about the Apple watch, which is less relevant to the input scenario that talks about Mike making a tree. The event generated by infilling GPT-2 is coherent with the scenario, but doesn't occur in the right order with respect to following two events. The event generated by EventBART, on the other hand, is the best in terms of the coherence and the temporality.

6.5 Learning Timex Knowledge

The temporal ordering and event infilling tasks correspond to information that we might expect to be encoded by our model pre-training. To test whether our models generalize to slightly more distant temporal phenomena, we examine whether they are able to capture the temporal relationships between timexes. This knowledge has been shown to be hard to learn in temporal relation extraction models [12].

Event 1	Tom died **in 1618**
Event 2	Jake died **in 1145**
Event 3	Mike died **in 1890**

Event 1	Durer went to a supermarket **at 7:52 pm**
Event 2	Durer bought a book at a shop **at 5:16 am**
Event 3	Durer took a photo in front of a museum **at 3:10 pm**

Figure 6.4: Examples of test input event sequences for timex evaluation. The appended timex in each event is highlighted. The first half of the figure is an example for sequences with "year" timex, with the names chosen randomly. The second half is an example for 12-hour clock time, whose event templates are also used in other types of timex except "Year".

6.5.1 Evaluation Setup

The timexes we examine here include years, months, weekdays, 24-hour clock time in "hour:minute" format and 12-hour clock time in "hour:minute am/pm" format. We evaluate the ability of our models to order events that are anchored with a timex in their arguments. To prepare the test input event sequences of a given type of timex, we first artificially make up a template event sequence with 3 typical daily events that have no temporal order relations. We then randomly sample 3 different timexes, e.g "June", "May", "July" for "Month", and append each of them to the events in the template sequence respectively with proper prepositions. At the end, 100 examples are created with this process for each type of timex. More concrete examples are shown in Figure 6.4. For "'Year" timex, we use dying events instead as the templates since those daily events co-occur less with years in natural language. For the

Architecture	Year		Month		Weekday		Hour:Minute (24)		Hour:Minute (12)	
	EM	Pairwise Acc.	EM	Pairwise Acc.	EM	Pairwise Acc.	EM	Pairwise Acc.	EM	Pairwise Acc.
Random	26.0	53.0	21.0	51.0	18.0	52.0	18.0	50.7	13.0	52.7
GPT-2	17.0	47.7	14.0	45.3	18.0	55.3	12.0	43.3	15.0	46.7
GPT-2 Large	24.0	51.3	16.0	47.7	19.0	57.7	9.0	41.7	11.0	48.7
EventBART	**94.0**	**97.7**	83.0	88.7	67.0	78.0	**88.0**	**93.3**	**67.0**	**78.7**
EventBART-indexed	89.0	96.0	**85.0**	**90.0**	**71.0**	**80.7**	84.0	90.7	65.0	78.0

Table 6.5: The results of temporal event ordering on the events anchored with various types of timex. The test data used here are length-3 sequences artificially made up with "die" events for the "Year" timex, and 3 typical daily events as shown in Figure 6.4 for other types of timex. The timexes of type "Year" are randomly sampled from 1000 to 2100. Our BART-based models significantly outperform the GPT-2 and random baselines, showing that they can capture useful timex-related knowledge.

baselines, here we use GPT-2 models to do the ordering by using the generation probability to rank all permutations of the input events.

6.5.2 Results

The results are shown in Table 6.5. First, we examine the results of the GPT-2 models. In general both the unsupervised GPT-2 (the medium model) and GPT-2 large perform worse than the random baseline, indicating that they have a limited ability to order timexes. Our BART-based models achieve stronger results. The results are strongest on years. For 12-hour clock time, even though it should be quite hard to tackle since the model has to link the temporal knowledge on "am' and "pm" with numerical comparisons, both of the BART-based models still performs significantly better the random baseline.

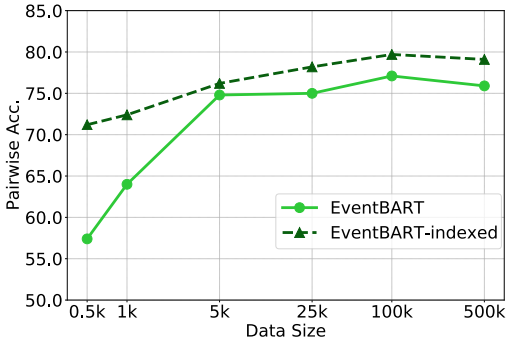


Figure 6.5: The pairwise accuracy of the two proposed BART-based event models on temporal event ordering task on CaTeRS when training with different number of event sequences from narrative documents. As the data size increases, the models are able to achieve a better performance.

6.6 The Effectiveness of Narrative Data

Here we investigate our model's scaling behavior with training data size. We train our BART-based models on varying size of the narrative temporal event sequences, and evaluate each model in a zero-shot manner on temporal event ordering task on CaTeRS dataset. Figure 6.5 shows that the performance of both of models improves as training data size increases. This demonstrates that the automatically extracted temporal event sequences are useful and diverse enough to help the models to learn temporal-related knowledge. The EventBART-indexed model is effective on surprisingly small amounts of data, but also scales well with data size; however, we observe a plateau in both models which motivated our decision to use 100k training sequences.

Chapter 7

Conclusion

This work presents a BART-based conditional generation model as well as a denoising autoencoder framework to learn temporal event knowledge, and addresses both temporal event ordering and event infilling tasks by pretraining on automatically collected data. By evaluating our model on the test data extracted from both CaTeRS and MCTaco datasets, we demonstrate that our model is able to perform temporal event ordering in a zero-shot manner, not fine-tuning on either of these datasets. Our model is also capable of generating coherent, temporally-sensible events to insert in seed event sequences. The demonstrated strengths of our model suggest that it can be an important component for schema learning and applied to other settings as well.

Bibliography

[1] Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. COMET: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1470. URL https://www.aclweb.org/anthology/P19-1470.

[2] Taylor Cassidy, Bill McDowell, Nathanael Chambers, and Steven Bethard. An annotation framework for dense event ordering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 501–506, Baltimore, Maryland, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/P14-2082. URL https://www.aclweb.org/anthology/P14-2082.

[3] Nathanael Chambers. Event schema induction with a probabilistic entity-driven model. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D13-1185.

[4] Nathanael Chambers and Dan Jurafsky. Unsupervised learning of nar-

rative event chains. In *Proceedings of ACL-08: HLT*, pages 789–797, Columbus, Ohio, June 2008. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P08-1090.

[5] Nathanael Chambers, Shan Wang, and Dan Jurafsky. Classifying temporal relations between events. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 173–176, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P07-2044.

[6] Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. Dense event ordering with a multi-pass architecture. *Transactions of the Association for Computational Linguistics*, 2:273–284, 2014. doi: 10.1162/tacl_a_00182. URL https://www.aclweb.org/anthology/Q14-1022.

[7] Fei Cheng and Yusuke Miyao. Classifying temporal relations by bidirectional LSTM over dependency paths. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–6, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-2001. URL https://www.aclweb.org/anthology/P17-2001.

[8] Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and

play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1edEyBKDS.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

[10] Quang Do, Wei Lu, and Dan Roth. Joint inference for event timeline construction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 677–687, Jeju Island, Korea, July 2012. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D12-1062.

[11] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. Allennlp: A deep semantic natural language processing platform. 2017.

[12] Tanya Goyal and Greg Durrett. Embedding time expressions for deep temporal ordering models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4400–4406, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1433. URL https://www.aclweb.org/anthology/P19-1433.

[13] Rujun Han, I-Hung Hsu, Mu Yang, Aram Galstyan, Ralph Weischedel, and Nanyun Peng. Deep structured neural network for event temporal relation extraction. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 666–106, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/K19-1062. URL https://www.aclweb.org/anthology/K19-1062.

[14] Rujun Han, Qiang Ning, and Nanyun Peng. Joint event and temporal relation extraction with shared representations and structured prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 434–444, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1041. URL https://www.aclweb.org/anthology/D19-1041.

[15] Parag Jain, Priyanka Agrawal, Abhijit Mishra, Mohak Sukhwani, Anirban

Laha, and Karthik Sankaranarayanan. Story generation from sequence of independent short descriptions. *CoRR*, abs/1707.05501, 2017. URL http://arxiv.org/abs/1707.05501.

[16] Bram Jans, Steven Bethard, Ivan Vulic, and Marie-Francine Moens. Skip n-grams and ranking functions for predicting script events. In *EACL*, 2012.

[17] Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 252–262, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1023. URL https://www.aclweb.org/anthology/N18-1023.

[18] Hirokazu Kiyomaru, Kazumasa Omura, Yugo Murawaki, Daisuke Kawahara, and Sadao Kurohashi. Diversity-aware event prediction based on a conditional variational autoencoder with reconstruction. In *Proceedings of the First Workshop on Commonsense Inference in Natural Language Processing*, pages 113–122, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-6014. URL https://www.aclweb.org/anthology/D19-6014.

[19] Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. Unsupervised machine translation using monolingual corpora only. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rkYTTf-AZ.

[20] Artuur Leeuwenberg and Marie-Francine Moens. Structured learning for temporal relation extraction from clinical records. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1150–1158, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/E17-1108.

[21] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL https://www.aclweb.org/anthology/2020.acl-main.703.

[22] X. Lu, Y. Tsao, S. Matsuda, and C. Hori. Speech enhancement based on deep denoising autoencoder. In *INTERSPEECH*, 2013.

[23] Aman Madaan and Yi-Ming Yang. Neural language modeling for contextualized temporal graph generation. *ArXiv*, abs/2010.10077, 2020.

[24] Aman Madaan, Dheeraj Rajagopal, Yiming Yang, Abhilasha Ravichander, E. Hovy, and Shrimai Prabhumoye. Eigen: Event influence generation using pre-trained language models. *ArXiv*, abs/2010.11764, 2020.

[25] Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. Machine learning of temporal relations. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 753–760, Sydney, Australia, July 2006. Association for Computational Linguistics. doi: 10.3115/1220175.1220270. URL `https://www.aclweb.org/anthology/P06-1095`.

[26] Ashutosh Modi. Event embeddings for semantic script modeling. In *CoNLL*, 2016.

[27] R. Mooney and G. DeJong. Learning schemata for natural language processing. In *IJCAI*, 1985.

[28] Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1098. URL `https://www.aclweb.org/anthology/N16-1098`.

[29] Nasrin Mostafazadeh, Alyson Grealish, Nathanael Chambers, James Allen, and Lucy Vanderwende. CaTeRS: Causal and temporal relation scheme for semantic annotation of event structures. In *Proceedings of the Fourth Workshop on Events*, pages 51–61, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/W16-1007. URL https://www.aclweb.org/anthology/W16-1007.

[30] Qiang Ning, Zhili Feng, and Dan Roth. A structured learning approach to temporal relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1027–1037, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1108. URL https://www.aclweb.org/anthology/D17-1108.

[31] Qiang Ning, Hao Wu, and Dan Roth. A multi-axis annotation scheme for event temporal relations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1318–1328, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1122. URL https://www.aclweb.org/anthology/P18-1122.

[32] Qiang Ning, Hao Wu, Rujun Han, Nanyun Peng, Matt Gardner, and Dan Roth. TORQUE: A reading comprehension dataset of temporal ordering questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1158–1172,

Online, November 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-main.88.

[33] Martha Palmer, Daniel Gildea, and Paul Kingsbury. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106, 2005. doi: 10.1162/0891201053630264. URL https://www.aclweb.org/anthology/J05-1004.

[34] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[35] Haoruo Peng and Dan Roth. Two discourse driven language models for semantics. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 290–300, Berlin, Germany, August 2016. Association for Computational Lin-

guistics. doi: 10.18653/v1/P16-1028. URL https://www.aclweb.org/anthology/P16-1028.

[36] Haoruo Peng, Snigdha Chaturvedi, and Dan Roth. A joint model for semantic sequences: Frames, entities, sentiments. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 173–183, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/K17-1019. URL https://www.aclweb.org/anthology/K17-1019.

[37] Karl Pichotta and J. Raymond Mooney. Learning statistical scripts with lstm recurrent neural networks. *AAAI*, pages 2800–2806, 2016.

[38] Karl Pichotta and Raymond J. Mooney. Learning statistical scripts with lstm recurrent neural networks. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2800–2806. AAAI Press, 2016.

[39] James Pustejovsky, José M. Castaño, Robert Ingria, Roser Saurí, Robert J. Gaizauskas, Andrea Setzer, Graham Katz, and Dragomir R. Radev. TimeML: Robust Specification of Event and Temporal Expressions in Text. In *New Directions in Question Answering*, 2003.

[40] James Pustejovsky, Patrick Hanks, Roser Saurí, Andrew See, Rob Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, and Marcia Lazo. The timebank corpus. *Proceedings of Corpus Linguistics*, 01 2003.

[41] Lianhui Qin, Vered Shwartz, Peter West, Chandra Bhagavatula, Jena D. Hwang, Ronan Le Bras, Antoine Bosselut, and Yejin Choi. Back to the future: Unsupervised backprop-based decoding for counterfactual and abductive commonsense reasoning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 794–805, Online, November 2020. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/2020.emnlp-main.58`.

[42] A. Radford, Jeffrey Wu, R. Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[43] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, M. Matena, Yanqi Zhou, W. Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

[44] Hannah Rashkin, Asli Celikyilmaz, Yejin Choi, and Jianfeng Gao. PlotMachines: Outline-conditioned generation with dynamic plot state tracking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4274–4295, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.349. URL `https://www.aclweb.org/anthology/2020.emnlp-main.349`.

[45] Julien Tourille, Olivier Ferret, Aurélie Névéol, and Xavier Tannier. Neural architecture for temporal relation extraction: A Bi-LSTM approach

for detecting narrative containers. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 224–230, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-2035. URL https://www.aclweb.org/anthology/P17-2035.

[46] Siddharth Vashishtha, Benjamin Van Durme, and Aaron Steven White. Fine-grained temporal relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2906–2919, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1280. URL https://www.aclweb.org/anthology/P19-1280.

[47] Marc Verhagen and James Pustejovsky. Temporal processing with the TARSQI toolkit. In *Coling 2008: Companion volume: Demonstrations*, pages 189–192, Manchester, UK, August 2008. Coling 2008 Organizing Committee. URL https://www.aclweb.org/anthology/C08-3012.

[48] Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. SemEval-2007 task 15: TempEval temporal relation identification. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 75–80, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/S07-1014.

[49] P. Vincent, H. Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, 2010.

[50] Su Wang, Eric Holgate, Greg Durrett, and Katrin Erk. Picking apart story salads. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1455–1465, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1175. URL `https://www.aclweb.org/anthology/D18-1175`.

[51] Su Wang, Greg Durrett, and Katrin Erk. Query-focused scenario construction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2712–2722, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1273. URL `https://www.aclweb.org/anthology/D19-1273`.

[52] Su Wang, Greg Durrett, and Katrin Erk. Narrative interpolation for generating and understanding stories. *CoRR*, abs/2008.07466, 2020. URL `https://arxiv.org/abs/2008.07466`.

[53] N. Weber, Niranjan Balasubramanian, and Nathanael Chambers. Event representations with tensor-based compositions. In *AAAI*, 2018.

[54] N. Weber, L. Shekhar, Niranjan Balasubramanian, and N. Chambers. Hierarchical quantized representations for script generation. In *EMNLP*, 2018.

[55] Noah Weber, Leena Shekhar, Niranjan Balasubramanian, and Nathanael Chambers. Hierarchical quantized representations for script generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3783–3792, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1413. URL `https://www.aclweb.org/anthology/D18-1413`.

[56] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/2020.emnlp-demos.6`.

[57] Lili Yao, Nanyun Peng, Ralph M. Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. Plan-and-write: Towards better automatic story-

telling. *CoRR*, abs/1811.05701, 2018. URL http://arxiv.org/abs/1811.05701.

[58] Wenlin Yao and Ruihong Huang. Temporal event knowledge acquisition via identifying narratives. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 537–547, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1050. URL https://www.aclweb.org/anthology/P18-1050.

[59] Xinyu Zhao, Shih ting Lin, and Greg Durrett. Effective distant supervision for temporal relation extraction. *ArXiv*, abs/2010.12755, 2020.

[60] Ben Zhou, Daniel Khashabi, Qiang Ning, and Dan Roth. "going on a vacation" takes longer than "going for a walk": A study of temporal commonsense understanding. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3363–3369, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1332. URL https://www.aclweb.org/anthology/D19-1332.

[61] Ben Zhou, Kyle Richardson, Qiang Ning, Tushar Khot, A. Sabharwal, and D. Roth. Temporal reasoning on implicit events from distant supervision. *ArXiv*, abs/2010.12753, 2020.